dydx — Calculate numeric derivatives and integrals

Description

`dydx` and `integ` calculate derivatives and integrals of numeric “functions”.

Quick start

For variables `y` and `x` corresponding to function $y = f(x)$, compute dy/dx using cubic splines and store result in `dy`

```
dydx y x, generate(dy)
```

Evaluate the integral of $f(x)$ using cubic splines

```
integ y x
```

Evaluate the integral using the trapezoidal rule

```
integ y x, trapezoid
```

Same as above, and generate variable `iy` containing integral evaluated for each value of `x`

```
integ y x, trapezoid generate(iy)
```

Menu

dydx

Data > Create or change data > Other variable-creation commands > Calculate numerical derivatives

integ

Data > Create or change data > Other variable-creation commands > Calculate numeric integrals

Syntax

Derivatives of numeric functions

`dydx` *yvar xvar* [*if*] [*in*] , `generate(`*newvar*`)` [*dydx_options*]

Integrals of numeric functions

`integ` *yvar xvar* [*if*] [*in*] [, *integ_options*]

dydx_options	Description
Main	
* `generate(`*newvar*`)`	store results in variable named *newvar*
`replace`	overwrite the existing variable
`double`	store new variable as `double`

* `generate(`*newvar*`)` is required.

integ_options	Description
Main	
`trapezoid`	use trapezoidal rule to compute integrals; default is cubic splines
`generate(`*newvar*`)`	store results in variable named *newvar*
`replace`	overwrite the existing variable
`double`	store new variable as `double`
`initial(`*#*`)`	initial value of integral; default is `initial(0)`

`by` and `collect` are allowed with `dydx` and `integ`; see **[U] 11.1.10 Prefix commands**.

Options for dydx

Main

`generate(`*newvar*`)` specifies that results be stored in a new variable. `generate()` is required.

`replace` specifies that if an existing variable is specified for `generate()`, it should be overwritten.

`double` specifies that the new variable in `generate()` be stored as `double`. If the `double` option is not specified, the variable is stored using the current type as set by `set type`, which is `float` by default.

Options for integ

Main

`trapezoid` requests that the trapezoidal rule [the sum of $(x_i - x_{i-1})(y_i + y_{i-1})/2$] be used to compute integrals. The default is cubic splines, which give superior results for most smooth functions; for irregular functions, `trapezoid` may give better results.

`generate(`*newvar*`)` specifies that results be stored in a new variable.

`replace` specifies that if an existing variable is specified for `generate()`, it should be overwritten.

`double` specifies that the new variable in `generate()` be stored as `double`. If the `double` option is not specified, the variable is stored using the current type as set by `set type`.

`initial(#)` specifies the initial condition for calculating definite integrals; see *Methods and formulas* below. The default is `initial(0)`.

Remarks and examples

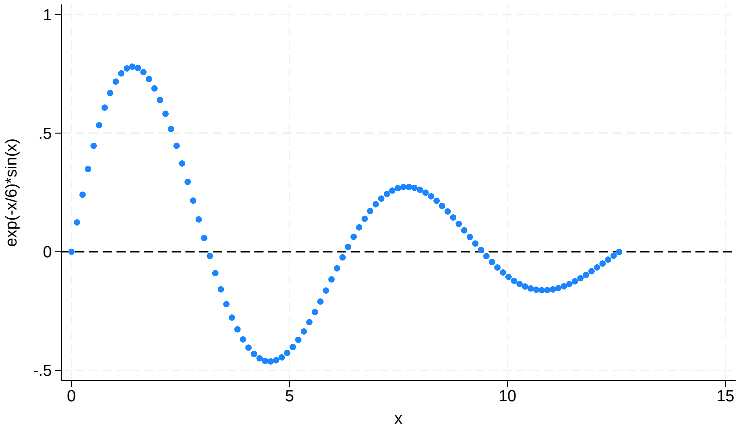
stata.com

`dydx` and `integ` lets you extend Stata's graphics capabilities beyond data analysis and into mathematics.

▷ Example 1

We graph $y = e^{-x/6}\sin(x)$ over the interval $[0, 12.56]$:

```
. range x 0 12.56 100
Number of observations (_N) was 0, now 100.
. generate y = exp(-x/6)*sin(x)
. label variable y "exp(-x/6)*sin(x)"
. twoway connected y x, connect(i) yline(0)
```

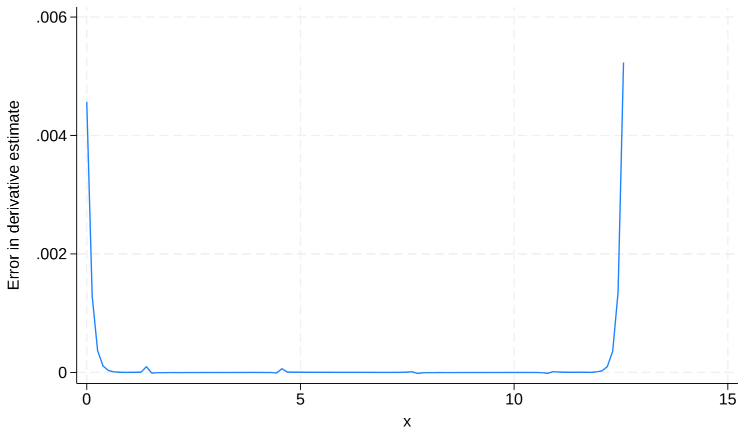


We estimate the derivative by using `dydx` and compute the relative difference between this estimate and the true derivative.

```
. dydx y x, gen(dy)
. generate dytrue = exp(-x/6)*(cos(x) - sin(x)/6)
. generate error = abs(dy - dytrue)/dytrue
```

The error is greatest at the endpoints, as we would expect. The error is approximately 0.5% at each endpoint, but the error quickly falls to less than 0.01%.

```
. label variable error "Error in derivative estimate"
. twoway line error x, ylabel(0(.002).006)
```

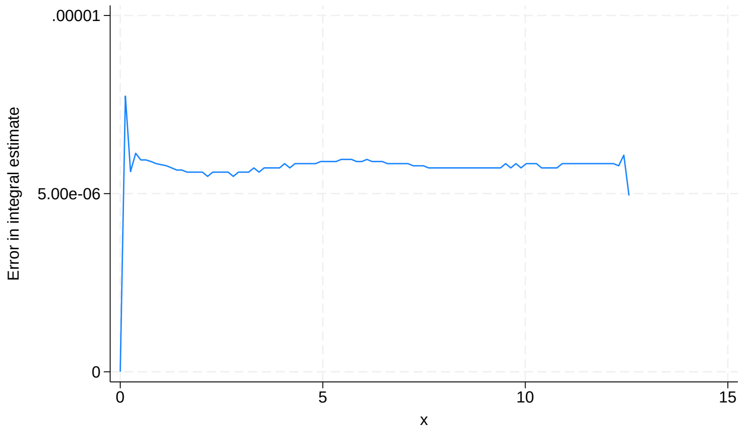


We now estimate the integral by using `integ`:

```
. integ y x, gen(iy)
number of points = 100
integral         = .85316397
. generate iytrue = (36/37)*(1 - exp(-x/6)*(cos(x) + sin(x)/6))
. display iytrue[_N]
.85315901
. display abs(r(integral) - iytrue[_N])/iytrue[_N]
5.811e-06
. generate diff = iy - iytrue
```

The relative difference between the estimate [stored in `r(integral)`] and the true value of the integral is about 6×10^{-6}. A graph of the absolute difference (`diff`) is shown below. Here error is cumulative. Again, most of the error is due to a relatively poorer fit near the endpoints.

```
. label variable diff "Error in integral estimate"
. twoway line diff x, ylabel(0(5.00e-06).00001)
```



◁

Stored results

`dydx` stores the following in `r()`:

Macros

`r(y)`	name of *yvar*

`integ` stores the following in `r()`:

Scalars

`r(N_points)`	number of unique x points
`r(integral)`	estimate of the integral

Methods and formulas

Consider a set of data points, $(x_1, y_1), \dots, (x_n, y_n)$, generated by a function $y = f(x)$. `dydx` and `integ` first fit these points with a cubic spline, which is then analytically differentiated (integrated) to give an approximation for the derivative (integral) of f.

The cubic spline (see, for example, Press et al. [2007]) consists of $n-1$ cubic polynomials $P_i(x)$, with the ith one defined on the interval $[x_i, x_{i+1}]$,

$$P_i(x) = y_i a_i(x) + y_{i+1} b_i(x) + y_i'' c_i(x) + y_{i+1}'' d_i(x)$$

where

$$a_i(x) = \frac{x_{i+1} - x}{x_{i+1} - x_i} \qquad b_i(x) = \frac{x - x_i}{x_{i+1} - x_i}$$

$$c_i(x) = \frac{1}{6}(x_{i+1} - x_i)^2 a_i(x)[\{a_i(x)\}^2 - 1] \qquad d_i(x) = \frac{1}{6}(x_{i+1} - x_i)^2 b_i(x)[\{b_i(x)\}^2 - 1]$$

and y_i'' and y_{i+1}'' are constants whose values will be determined as described below. The notation for these constants is justified because $P_i''(x_i) = y_i''$ and $P_i''(x_{i+1}) = y_{i+1}''$.

Because $a_i(x_i) = 1$, $a_i(x_{i+1}) = 0$, $b_i(x_i) = 0$, and $b_i(x_{i+1}) = 1$. Therefore, $P_i(x_i) = y_i$, and $P_i(x_{i+1}) = y_{i+1}$. Thus, the P_i jointly define a function that is continuous at the interval boundaries. The first derivative should be continuous at the interval boundaries; that is,

$$P_i'(x_{i+1}) = P_{i+1}'(x_{i+1})$$

The above $n-2$ equations (one equation for each point except the two endpoints) and the values of the first derivative at the endpoints, $P_1'(x_1)$ and $P_{n-1}'(x_n)$, determine the n constants y_i''.

The value of the first derivative at an endpoint is set to the value of the derivative obtained by fitting a quadratic to the endpoint and the two adjacent points; namely, we use

$$P_1'(x_1) = \frac{y_1 - y_2}{x_1 - x_2} + \frac{y_1 - y_3}{x_1 - x_3} - \frac{y_2 - y_3}{x_2 - x_3}$$

and a similar formula for the upper endpoint.

`dydx` approximates $f'(x_i)$ by using $P_i'(x_i)$.

`integ` approximates $F(x_i) = F(x_1) + \int_{x_1}^{x_i} f(x)\,dx$ by using

$$I_0 + \sum_{k=1}^{i-1} \int_{x_k}^{x_{k+1}} P_k(x)\,dx$$

where I_0 (an estimate of $F(x_1)$) is the value specified by the `initial(#)` option. If the `trapezoid` option is specified, `integ` approximates the integral by using the trapezoidal rule:

$$I_0 + \sum_{k=1}^{i-1} \frac{1}{2}\,(x_{k+1} - x_k)(y_{k+1} + y_k)$$

If there are ties among the x_i, the mean of y_i is computed at each set of ties and the cubic spline is fit to these values.

Acknowledgment

The present versions of dydx and integ were inspired by the dydx2 command written by Patrick Royston of the MRC Clinical Trials Unit, London, and coauthor of the Stata Press book *Flexible Parametric Survival Analysis Using Stata: Beyond the Cox Model*.

Maria Gaetana Agnesi (1718–1799) was an Italian mathematician and philosopher.

Born in Milan into a wealthy family, she was recognized as a child prodigy. At age nine, she published a detailed argument in Latin on the importance of education for women. At age 15, her father, a mathematics professor at the University of Bologna, presented her talents in language and philosophical reasoning to Bologna's intellectual elite. Uncomfortable with public life, she educated her twenty siblings and published on mathematics. After her father's death in 1752, she studied theology and devoted the rest of her life to helping the poor, homeless, and sick.

Agnesi's best known work, *Instituzioni analitiche ad uso della gioventù italiana* (*Analytical Institutions for the Use of Italian Youth*), written in 1748, helped develop the analysis of finite quantities and infinitesimals. At the time, it was hailed as the best introduction to calculus. The work also discussed an asymptotic curve that, because of mistranslation, would come to be known as the "Witch of Agnesi". In 1750, Pope Benedict XIV appointed her to the chair of mathematics and natural philosophy at Bologna, though she never served.

In addition to being recognized as an important mathematician, Agnesi is revered in the Basilica of San Nazaro in Milan. At her death, she was mourned by radical authors as a proponent for women's rights and by the Catholic faithful as a symbol of personal piety.

Reference

Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. 2007. *Numerical Recipes: The Art of Scientific Computing*. 3rd ed. New York: Cambridge University Press.

Also see

[D] **obs** — Increase the number of observations in a dataset

[D] **range** — Generate numerical range